AD-A079 330 ARMY PERSONNEL RESEARCH OFFICE WASHINGTON DC F/G 5/9
A GENERAL COMPUTER SIMULATION FOR CONDUCTING ALLOCATION EXPERIM--ETC(U)
FEB 67 E NIEHL
UNCLASSIFIED APRO-RM-67-1 NL

END
DATE FILMED
2-80
DDC

Research Memorandum 67-1

A GENERAL COMPUTER SIMULATION FOR CONDUCTING ALLOCATION EXPERIMENTS

FEBRUARY 1967

U. S. ARMY PERSONNEL RESEARCH OFFICE

Army Project Number
2J024701A723

CMS c-13

Research Memorandum 67-1

A GENERAL COMPUTER SIMULATION FOR CONDUCTING ALLOCATION EXPERIMENTS

by Elizabeth Niehl

Richard C. Sorenson, Task Leader

APRO-RM-67-1

Submitted by:
Cecil D. Johnson
Chief, Statistical Research
and Analysis Laboratory

Approved by:
J. E. Uhlaner
Director
USAPRO Laboratories

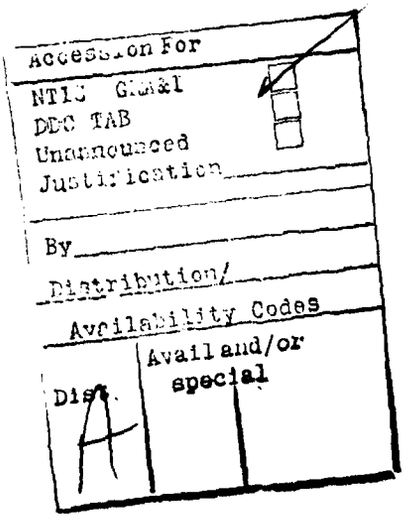
February 1967

040 650

PREFACE

The present Research Memorandum presents a general computer program developed in the Statistics Research and Analysis Laboratory of the U. S. Army Personnel Research Office for use in the analysis of simulated manpower information systems problems having common aspects.

With feasible modifications, the program is applicable to a number of major studies of manpower allocation policy and procedures.



A GENERAL COMPUTER SIMULATION FOR CONDUCTING ALLOCATION EXPERIMENTS

In studies of the Army's personnel systems, the effects of proposed alternative policies and procedures must be evaluated. In lieu of experimentation with the real system--trial implementation of proposed changes, for example--the U. S. Army Personnel Research Office has developed a series of models representing the system. Some of these models have been computerized and made dynamic--computer simulations, in other words.

A number of experiments have been performed in U. S. APRO's Statistical Research and Analysis Laboratory in which simulated samples of enlisted men were generated and treated experimentally. Optimal allocation to job clusters associated with the eight Army aptitude areas on the basis of performance estimates linearly transformed from scores on the Army Classification Battery[1,2] was an important feature of the experiments. In one such study, the transformation consisted of least squares regression coefficients based on all eleven tests of the ACB rather than on computationally simplified two-test composites--the aptitude area scores. It was found that the sum of the performance estimates for the jobs to which the men were assigned was substantially increased by the experimental procedure. This "allocation sum" has also been studied as a function of the metric in which the performance estimates were expressed[3]. For example, performance estimates based on 100-unit equal interval scales were found to yield higher overall estimated performance than estimates based on ordinal scales constructed by ranking performance on each job over individuals or by ranking each individual over jobs. Each of these scales was found to be superior, however, to equal interval scales with only 10 units. In studies with a still different orientation, predicted changes in overall performance were examined when different screening or allocation policies for enlisted men were simulated. Alternative policies consisted of variations in requisites for assignment to occupational areas in terms of estimated performance levels. Estimates were based on scores on the Armed Forces Qualification Test (AFQT), the Army Qualification Battery (AQB), the Army Classification Battery (ACB), or some specified combination of scores.

---

[1] Sorenson, Richard C. Optimal allocation of enlisted men--Full regression equations vs aptitude area scores. USAPRO Technical Research Note 163, November 1965.

[2] Sorenson, Richard C. Effect of modification in operational procedures on an optimal personnel allocation system. Paper presented at the Convention of the American Psychological Association, September 1965. Abstract appeared in American Psychologist 20:7, 1965.

[3] Sorenson, Richard C. The effect of metric changes on resource allocation decisions. Proceedings of the U. S. Army Operations Research Symposium, Fort Monmouth, New Jersey, March 1966.

Each of these experiments has required the preparation of a separate and lengthy computer program. Yet, there was considerable similarity in the programs. To facilitate experiments involving the generation of simulated samples, U. S. APRO research scientists have developed a general simulation program which may be used to evaluate a variety of alternative policies. The program, designed to perform the many operations common to experiments of this kind, can easily be modified to handle computations specific to different experiments.

GENERAL OPERATIONS

The general part of the program begins with the automatic generation of a vector of random normal deviates for every individual; on this vector are performed a series of linear transformations of the form $\underline{V} = \underline{U}\,T + \underline{M}$. T may be a matrix of least squares regression coefficients for obtaining performance estimates $\underline{V}$ from a set of predictor scores $\underline{U}$. $\underline{M}$ are the additive constants to yield estimates with specified means. A special purpose for which T is used in simulation studies is to transform random normal deviates which have an expected covariance matrix of identity into variates with an expected covariance matrix characteristic of the population under investigation. The user specifies the series of linear transformations required to generate a particular sample by first inputting the matrices (and associated mean vectors) and then referencing these matrices on special transformation cards, which are input in the order of the transformations to be performed.

SPECIALIZED SUBROUTINES

To perform non-linear transformations on the score vectors generated for each individual, specially written parameter subroutines may be easily incorporated in the sequence of operations under control of the main program. Modification and recompilation is required only for a short executive subprogram, not for the main program. The parameter subroutines are assigned integer names and are called by listing these integers, in the order in which the subroutines are to be performed, on the cards which define the sequence of linear transformations. These subroutines are also used to determine whether the scores being generated for a given individual are consistent with sample characteristics defined for a particular investigation. As the operations specified on each transformation card are completed, a special parameter, which may have been set to indicate rejection of the given individual by any of the parameter subroutines, is automatically sensed by the main program. Thus, tests for individual acceptance or rejection may be performed repeatedly and at any stage in the computations.

The result of the sequence of operations specified on the transformation cards is the construction of a $\nu \times \rho$ matrix of performance measures; the ρ columns correspond to job categories under investigation, and the ν

rows represent the individuals in the sample. Based on this matrix, each of the ν individuals is assigned to one of the ρ jobs in such a way that required quotas for the different jobs are filled, and the sum of the performance estimates for the jobs assigned to the ν individuals is maximal. The solution for this optimal allocation may be considered a special case of Kuhn's[1] solution to the transportation problem.

After optimal allocation, the response vectors for the ν individuals are regenerated to compute statistics that are functions of the job to which each individual is assigned. The programmer specifies the new computations by inputting a second set of transformation cards of the type input before allocation. (To eliminate unnecessary computations, special codes on the transformation cards can be used to restrict arithmetic operations to those involving the assigned jobs.) The same sequence of random numbers is automatically generated both before and after allocation so that the simulated performance measures will represent the same individuals.

The feature of specifying alternative sequences of linear transformations and subroutines as parameters may be considered a compiler for individual response simulation. Unfortunately, incorrect use of this part of the program will not produce simple error diagnostics, but may generate meaningless results. To insure that the same individuals are accepted into the sample before and after allocation, the user must specify the correct sequence of transformations and subroutine calls. In addition, care must be taken to maintain the correct order of simulated scores which represent tests of interest. For example, the transformation $\underline{V} = \underline{U}\,T$, which can be specified on a transformation card by equating an integer to the order in which the appropriate matrix was input, will operate only on the first α elements in the first working storage area $\underline{U}$ and will replace only the first β elements of the second working storage area $\underline{V}$; α and β are the number of rows and columns, respectively, of the transformation matrix T. Scores, then, which are to be modified by reference to parameter subroutines, but not by post-multiplication by one of the transformation matrices, must be represented by elements subsequent to those altered by linear transformations. Accuracy of results can be checked by setting a special debug parameter, which will produce detailed print-out of computations intermediate to the final results.

THE COMPUTATION PROCEDURE

The program operates in four parts: I, input and initialization required before sample generation; II, simulation of individual response vectors, optimal allocation, and regeneration of response vectors; III, computations and output based on the ν individuals in a sample, plus

[1] Kuhn, H. W. The Hungarian method for the assignment problem. Naval Research Logistic Quarterly, 2:1, 1955.

initialization required for generation of a new sample; and IV, computations and output based on all samples generated. Reference to the flow diagram, Figure 1, may clarify this sequence of operations.

To perform operations consistent with the four parts of the main program, four entry points must be written into every parameter subroutine. To transfer control to parts I, III, and IV, the main program automatically calls all subroutines listed on the transformation cards in order of increasing size of integer names assigned to the subroutines. (During execution of part II, the subroutines are called in the order listed on the transformation cards.) Dummy instructions ("returns") are written into entries for any of the four parts not used by a subroutine.

INPUT OPERATIONS

Operations which constitute Part I consist of the input of transformation matrices and associated mean vectors (box 1), input of the vector of starting numbers to initialize random number generation (box 2), input of the number of individuals to be simulated and other parameters defining characteristics of each sample (box 3), input of the transformation cards (box 6) and σ, the number of samples to be simulated (box 7). To make it convenient to generate δ different sets of samples during one computer run, each set being defined by different lists of transformation cards, the parameter δ is input (box 4). All transformation matrices required for the δ sets of samples are input simultaneously under control of the first part of the program (box 1). Any of these matrices may be used by any of the δ different problems by entering an integer which corresponds to the order in which the given matrix was input into the appropriate column of a transformation card.

The input of parameters of data required for execution of any of the parameter subroutines is represented by box 8. This input is called in order of increasing size of the integer-named subroutines. Any additional initialization required for subsequent operation of the subroutines is also performed within this first entry.

SIMULATION OPERATIONS

Part II of the program is illustrated in boxes 9-27 of the flow diagram. In box 9, the starting vector $\underline{A}_{sn}$, which controls the generation of random numbers, is set equal to the vector input to represent the first individual of sample 1 ($\underline{A}_{11}$ in box 2). This vector automatically changes as random numbers are generated for each individual (boxes 21 and 27). Box 16 refers to the case where the responses being generated for individual i have been found unacceptable by one of the parameter subroutines; consequently, another vector, $\underline{A}_{sn}$, is generated from which a

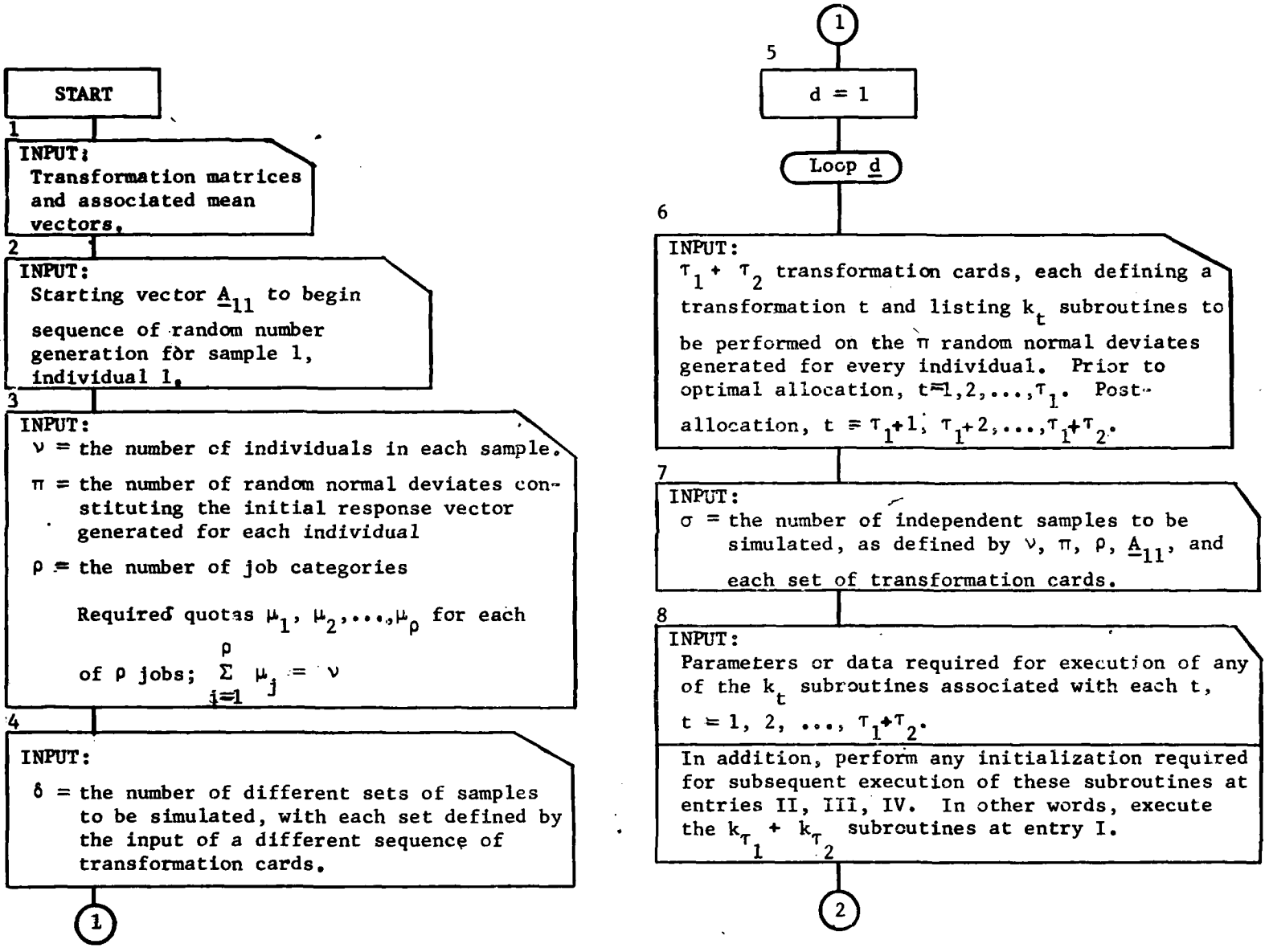


Figure 1. Flow diagram of General Program for Conducting Optimal Allocation Experiments

PART I - Input Operations

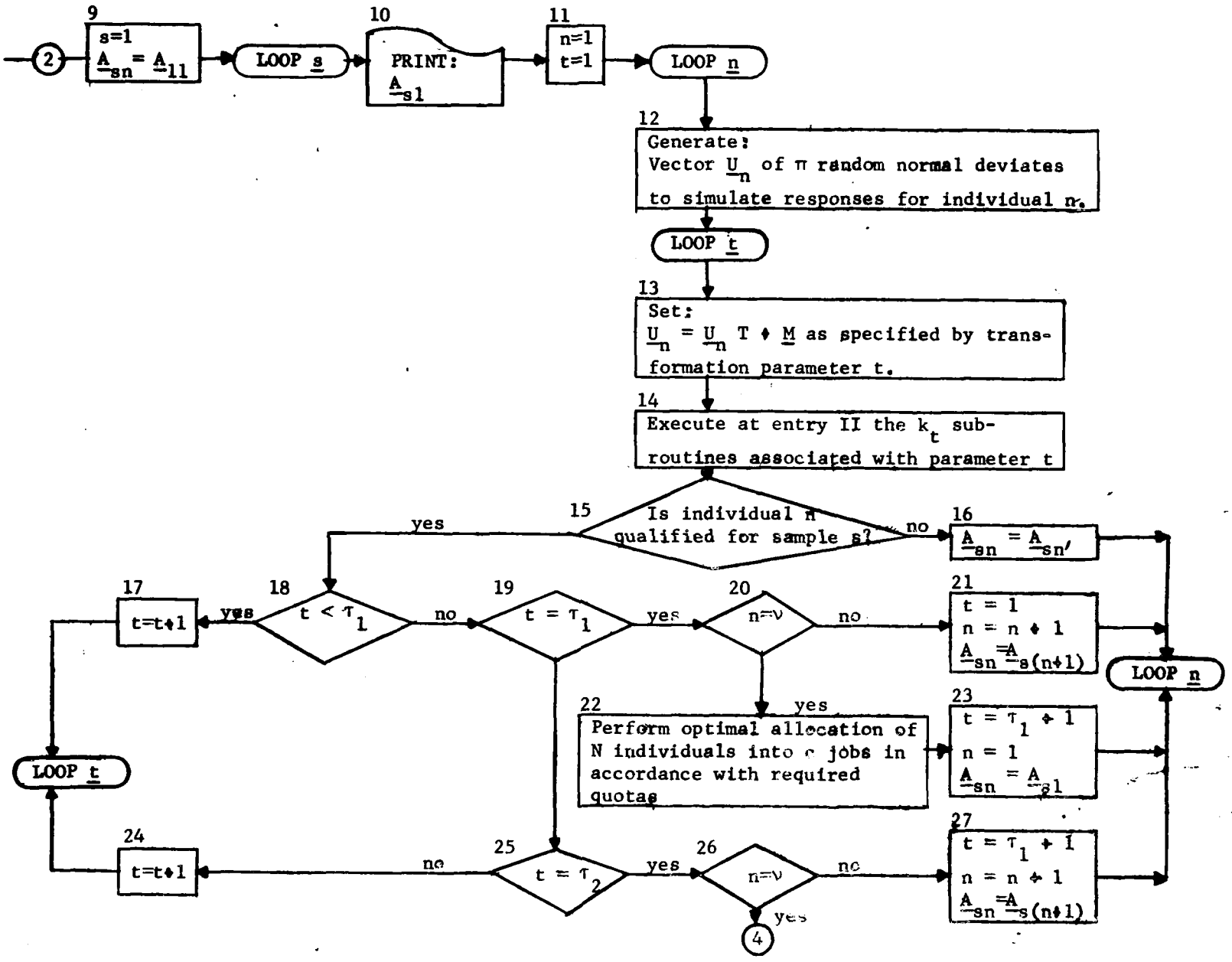


Figure 1. (continued) PART II - Simulation Operations

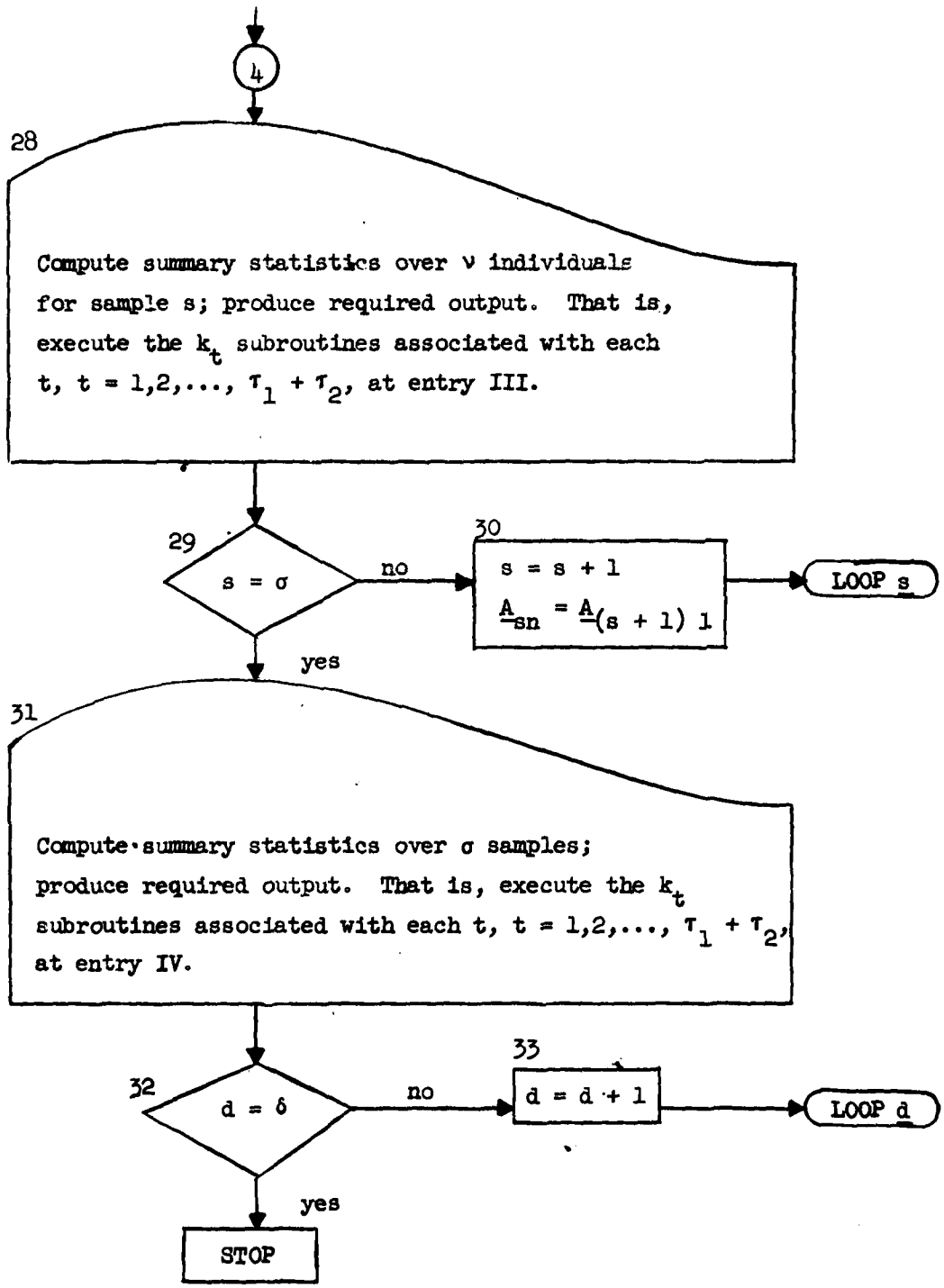


Figure 1 (continued).

PART III - Computations and Output (ν individuals) and

PART IV - Computations and Output (σ samples)

new set of scores representing the n^{th} individual can be constructed. To insure that the same set of individual responses are generated before and after allocation, the starting vector for individual 1 of sample s, $\underline{A}_{s1}$, is saved for the re-initialization shown in box 23. The starting vector which defines the beginning of each new sample is automatically printed out by the main program (box 10) so that simulation can be continued from any point of unanticipated termination of production.

Linear transformations and parameter subroutines specified by the first τ_1 transformation cards are performed on each individual response vector of random normal deviates (boxes 13 and 14) until the matrix consisting of c performance measures for ν individuals has been constructed. Execution of the $\tau_1 + \tau_2$ transformation cards is under control of the index t, as shown in boxes 17-19, 21, 23-25, 27. Box 15 illustrates the feature of examining the qualifications of each individual every time the computations specified on a transformation card are completed.

COMPUTATIONS AND OUTPUT

Parts III (box 28) and IV (box 31) represent the automatic calls to the parameter subroutines to perform operations over the ν individuals in a sample and then over the σ samples respectively. The final loop δ determines whether a new set of transformation cards will be input (at box 6) to define additional samples. The new samples will be based on the same set of parameters (ν, π, ρ, boxes 1-4) which defined the previous samples, with differences occurring only as specified on the new transformation cards. Note that the starting vector which begins the generation of random numbers is reset to $\underline{A}_{11}$ (box 9), so that equivalent sets of individual responses will be generated for each of the δ sets of samples.

MANUAL FOR USE OF GENERAL COMPUTER PROGRAM FOR SIMULATION EXPERIMENTS

INPUT FORMAT

iter 1 i j (Format 1XA6,I4,I2)

i is an integer which occupies columns 8-11 and specifies the number of complete sets of data and parameters to be input. To debug new subroutines, detailed print-out of individual score vectors may be obtained by setting j = 1, columns 12-13; otherwise, j = 0.

TRANSFORMATION MATRICES

1j, 2j,...,ij,... (Format 10(I1,I1,A1))

Beginning in column 1, a two-digit integer ij, which may be followed by a comma, is listed for every matrix input. i = 1 for the first matrix input and is increased by one for each successive matrix; the maximum i = nmat ("nmat" is a parameter set within the main program to determine the total number of transformation matrices to be input). If a vector of means associated with a given transformation is to be input, j = 1; otherwise, j = 0.

α β (Format 2I2)

MATRIX X (Format 1X, 9A8)

Transformation Matrix (Format 9F8.4)

For every ij listed, one card is input defining the order of the matrix, followed by a second card containing any alphanumeric name for the matrix, followed by the matrix itself. α (columns 1-2) is the number of rows, and β (columns 3-4) is the number of columns. Elements of the matrix are input row by row with the format 9F8.4.

MATRIX XM (Format 1X, 9A8)

Means Following the transformation matrix associated with a given ij, if j = 1, an additional alphanumeric card identifies the vector of means. The means begin in the next card with format 9F8.4.

Starting Vector

The γ = 24 octal digits which begin the generation of γ independent sequences of random numbers are input on 3 cards with format 9Ø8 for the C.D.C. computer (and on 4 cards with format 6Ø12 for the IBM 7094[5] at NBS).

---

[5] Commercial designations are given in the interest of precision in specifying the equipment used, and do not constitute official indorsement by the Army.

$\nu\ \pi\ \rho$ (Format I4,2I2)

ν (columns 1-4) is the number of individuals in the sample.

π (columns 5-6) is the number of independent random normal deviates generated for each individual to initiate the simulation of ρ performance measures.

ρ (columns 7-8) is the number of job categories on which optimal allocation is to be based.

$\mu_1\ \mu_2 \ldots \mu_\rho$ (Format 14I5)

Required quotas for each of ρ jobs: $\sum_{j=1}^{\rho} \mu_j = \nu$.

iter 2 = δ (Format 1XA6,I4)

δ specifies the number of complete sets of parameter cards and data which are to follow, all of which will be controlled by the set of cards previously input.

TRANSFORMATION CARDS

$\tau_1\ \tau_2$ (Format 2I2)

τ_1 cards will follow specifying transformations and subroutines to be performed on the vector of π normally distributed standard scores generated for individual n at working storage $\underset{\sim}{U}$, preceding optimal allocation. The next τ_2 cards specify the transformations and subroutines to be performed on the same individual response vector regenerated at $\underset{\sim}{U}$ following optimal allocation.

The post-multiplication of a response vector located at working storage $\underset{\sim}{U}$ (or $\underset{\sim}{V}$) by any transformation matrix input (T_I), the addition of the vector of means associated with the given transformation ($\underset{\sim}{M}_I$), and the storage of the result in working storage $\underset{\sim}{\tilde{V}}$ (or $\underset{\sim}{\tilde{U}}$) can be specified on any of the $\tau_1 + \tau_2$ transformation cards by the following codes:

$$12,I \Rightarrow \underset{1x\beta_I}{\tilde{V}} = \underset{1x\alpha_I}{\tilde{U}}\ \underset{\alpha_I \times \beta_I}{T_I} + \underset{1x\beta_I}{M_I}$$

$$21,I \Rightarrow \underset{1x\beta_I}{\tilde{U}} = \underset{1x\alpha_I}{\tilde{V}}\ \underset{\alpha_I \times \beta_I}{T_I} + \underset{1x\beta_I}{M_I}$$

Different T_I and $\underline{M}_I$ are referenced by setting I to any integer from 1 to nmat corresponding to the order in which the transformation matrices are input. If no vector of means has been imput for T_I, $\underline{M}_I = 0$. Multiplications are performed on only the first α_I elements of $\underset{\sim}{\underline{U}}$ (or $\underset{\sim}{\underline{V}}$), and the result changes only the first β_I elements of $\underset{\sim}{\underline{V}}$ (or $\underset{\sim}{\underline{U}}$); α_I and β_I are the number of rows and columns of T_I.

γ scores can be relocated from one working storage area to the other by setting I = 0. That is,

$$12,0 \Rightarrow \underset{1x\gamma}{\underset{\sim}{\underline{V}}} = \underset{1x\gamma}{\underset{\sim}{\underline{U}}}$$

$$21,0 \Rightarrow \underset{1x\gamma}{\underset{\sim}{\underline{U}}} = \underset{1x\gamma}{\underset{\sim}{\underline{V}}}$$

Following allocation over jobs j = 1, 2, . . ., ρ, a statistic, which is a function of the job j to which each individual is assigned, may be computed and stored in x ($\underset{\sim}{\underline{W}}(49)$). That is,

$$10,I \Rightarrow \underset{1x1}{\underset{\sim}{x}} = \underset{1x\alpha_I}{\underset{\sim}{\underline{U}}} \; \underset{\alpha_I x 1}{\underline{T}_{Ij}} + \underset{1x1}{m_{Ij}}$$

$$20,I \Rightarrow \underset{1x1}{\underset{\sim}{x}} = \underset{1x\alpha_I}{\underset{\sim}{\underline{V}}} \; \underset{\alpha_I x 1}{\underline{T}_{Ij}} + \underset{1x1}{m_{Ij}}$$

where $\underline{T}_{ij}$ is the j-th column of T_I and m_{Ij} is the j-th element of $\underline{M}_I$. If no transformation matrix is to be used, the j-th element of $\underset{\sim}{\underline{U}}$ (or $\underset{\sim}{\underline{V}}$) can be relocated in $\underset{\sim}{x}$ by setting I = 0:

$$10,0 \Rightarrow \underset{\sim}{x} = \underset{\sim}{u}_j$$

$$20,0 \Rightarrow \underset{\sim}{x} = \underset{\sim}{v}_j$$

PARAMETER SUBROUTINES

To perform additional kinds of computations on scores in $\tilde{U}$, $\tilde{V}$, or $\tilde{x}$, a list of subroutines may follow the transformation codes on any of the $t_1 + t_2$ transformation cards. Each subroutine is equated to some integer between 1 and 99 and is listed in the order of required execution. Beginning in column 5 with format 5(A1,I2), each subroutine is introduced by a comma, with the following integer name occupying two columns. As many as 5 subroutines may be referenced on a given card. To illustrate, suppose the post-multiplication of scores in $\tilde{U}$ by the second matrix input is required, to be followed by the execution of subroutines 3, 98, and 1, which have been especially written to operate on the result in $\tilde{V}$. The transformation card would appear as:

12,2,03,98,01

During simulation of scores for a given individual, a reject parameter is evaluated each time the processing defined by a transformation card is completed. Depending on the value of this parameter, scores for the individual are either retained for further processing or are treated as inappropriate for a given sample and eliminated. If it is desired to subdivide a list of subroutines so that the reject parameter will be evaluated after the execution of only one or two, say, of the subroutines, additional transformation cards defining the subroutine subsets are inserted, but with columns 1, 2, and 4 set to zero. To illustrate, suppose the γ scores in $\tilde{V}$ are to be transferred to $\tilde{U}$, to be followed by execution of subroutines 3, 98, and 1, written to operate on scores in $\tilde{U}$. The reject parameter, modified by subroutines 3 and 1, say, is to be evaluated after the execution of 3 and 1. The appropriate two transformation cards would appear as:

21,0,03

00,0,98,1

NUMBER OF SAMPLES

iter 1 = i (Format 1XA6,I4)

i is an integer which occupies columns 8-11 and specifies the number of independent samples to be simulated as defined by all previous parameter cards input.

INPUT CALLED BY SURBOUTINES

The input of parameters and data may be required for the execution of subroutines listed on the transformation cards. Inputs called by different subroutines is arranged by increasing order of the integers which name the subroutines.

DEFINITION OF DIMENSION STATEMENTS AS LISTED IN THE COMPUTER PROGRAM

KO(N,LC): Matrix of fixed-point performance estimates for N individuals on each of LC jobs. Currently (February 1967), N = 1000 and LC = 8.

IROW(N): Vector containing the job j, j = 1, 2, . . . , LC, to which each of N individuals is assigned.

KQ1(LC), KQ2(LC):

Storage areas for the required quotas for each of LC jobs.

Z(AREA): Storage area for all transformation matrices, thus requiring that

$$AREA \geq \sum_{m=1}^{NMAT} (NA_m \times MA_m),$$

where the order of each matrix input is NA (rows) x MA (columns), and the total number of matrices input is NMAT.

Currently, AREA = (23 x 23) + (11 x 8) + (11 x 8) = 705.

ZM(TMA): Storage area for means associated with each transformation matrix, with $TMA \geq \sum_{m=1}^{NMAT} MA_m$

Currently, TMA = 23 + 8 + 8 = 39.

U(S)[a], V(S), W(2S + 1), with U(1) = W(1) and V(1) = W(S + 1):

---

[a] The storage area represented here by S is represented in the description of the program by Y.

Working storage areas used for simulating a maximum of S individual responses.

Currently (February 1967), S = 24.

IISV(S), ISV(S), JSV(S):

Areas used for storing the starting vector of S octal digits called by subroutine RAND.

BBB(S) (required for the IBM 7094):

Working storage area used by subroutine RAND.

VFOR(9) (for CDC) or VFOR(12) (for IBM):

Area used for input of variable format.

IS(NTRAN), JS(NTRAN,5), KN(NTRAN), KN(NTRAN,6), LISI(NSUB):

Storage areas used for setting up the sequence of transformations and subroutines used in simulating responses for each individual. NTRAN is the maximum number of transformation cards which can be input for a given simulation experiment. NSUB is the maximum number of different parameter subroutines which can be listed on transformation cards for the experiment. Currently, NTRAN = 8 and NSUB = 10.

NA(NMAT), MA(NMAT), NT(NMAT):

Areas used for addressing transformation matrices and associated vectors of means. Currently, NMAT = 3.

For the most efficient utilization of computer space, the main program can be recompiled so that N, LC, AREA, TMA, and NMAT are consistent with requirements for specific projects. The statement "NMAT = 3," which is within the first part of the main program, will also have to be changed. Changes in S, NTRAN, and NSUB will not result in appreciable saving of space. If changes are required for S, NTRAN, and NSUB, additional changes must be made in statements of the main program.

PROGRAMMING REQUIREMENTS FOR PARAMETER SUBROUTINES

Any subroutine written to handle computations specific to different projects must be associated with an integer name, and the first parameter of every subroutine must be "JSUB". Thus, "SUBROUTINE SUB J (JSUB, ...)", where J is any integer, would appropriately begin a subroutine. In addition, the first statement of any subroutine must be "GO TO (a, b, c, d), JSUB", a, b, c, and d not all necessarily different.

All subroutines listed on the transformation cards are automatically called in four different parts of the main program. Corresponding to each part, JSUB is set to 1, 2, 3, or 4 to permit execution of different types of subroutine operations.

Before beginning response simulation for any sample of ν individuals, the main program sets JSUB = 1 and transfers control to statement a of each subroutine in order of increasing size of the integers naming the subroutines. Any initialization or calls for input of data or parameters required for subsequent operations must be written into this "part a" of each subroutine. If no such initialization is required, statement a will be a "RETURN".

During the simulation of individual score vectors in the second part of the main program, JSUB = 2; for each individual in every sample, control passes to statement b of each subroutine according to the order transformations and subroutines are listed on the transformation cards. The main function of subroutine sections beginning at statement b, then, will be to operate on scores in working storage areas $\tilde{U}$, $\tilde{V}$, or $\tilde{x}$ or to construct KO, the $\nu \times \rho$ matrix of scores used by subroutine OTT for optimal allocation.

After all computing to be performed on ν individuals has been completed, the main program sets JSUB = 3, and control is transferred to statement c of each subroutine, again in order of increasing size of the integers naming the subroutine. "Part c" of each subroutine must be written to perform any summary computations based on all ν individuals for the sample just completed, and to perform any initialization required for generating the next sample. If no such computations are required, statement c must be a "RETURN".

When simulation of the total of σ samples has been completed, the main program sets JSUB = 4, and control is transferred to statement d of each subroutine, again according to the order of the subroutine numbers. Any summary computations or output based on the completed set of σ samples or any initialization required before parameters defining a new set of σ samples are input must be written beginning at statement d. Otherwise, statement d must be a "RETURN".

MODIFICATION OF SUBROUTINE EXSUB

The call statement for any of the parameter subroutines listed on the transformation cards occurs in the executive subroutine. Each time, then, that a new subroutine is prepared for a specific project, "Subroutine EXSUB" must be recompiled.

At the time for execution of any of the subroutines listed on the transformation cards, the main program automatically sets a parameter "KM" within "EXSUB" to the integer name of the subroutine. Any test which recognizes that KM equals this integer name, J, say, and is followed by the execution of a statement calling "SUBROUTINE J", may be written into the executive routine. One convenient method would be to extend the length of the computed go-to statement (currently, "GO TO (1,2,3,4,5),KM"), making the j-th parenthesized integer equal to J such that, when KM = J, control will be transferred to statement J, and subroutine J will be executed. Care must be taken to insert any new subroutine call statements to operate within the range of the do-loop which terminates at statement (currently labeled) 25.

All regions likely to be referenced by specific subroutines are in the COMMON area for both the main program and EXSUB. Those regions most likely to be needed as parameters in any of the subroutines are defined as follows:

KO — the $\nu \times \rho$ matrix of fixed-point scores used by subroutine OTT for optimal allocation.

U and V — working storage areas beginning at $\tilde{W}(1)$ and $\tilde{W}(25)$, respectively, used for the post-multiplication of an individual score vector by a transformation matrix and possible subsequent addition of a mean vector.

LC (equivalent to ρ) — a single score selected from the ρ scores simulated for a given individual to correspond to j, $j = 1,2,\ldots,\rho$, the job to which the individual is assigned.

N — the number of individuals in a given sample.

I — index for the number of individuals: $I = 1,2,\ldots,N$

ITER3 (equivalent to S) — the number of samples of "N" individuals to be simulated for each set of transformation cards input.

IT3 — index for the number of samples: $IT3 = 1,2,\ldots,ITER3$